UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/752,121	12/29/2000	Julio Estrada	LOT9-2000-0023 US1	8745

7590 01/12/2005

Stephen T. Keohane, Esq.
Lotus Development Corporation
55 Cambridge Parkway
Cambridge, MA 27085

EXAMINER	
BLACKWELL, JAMES H	

ART UNIT	PAPER NUMBER
2176	

DATE MAILED: 01/12/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/752,121	ESTRADA ET AL.	
	Examiner	Art Unit	
	James H Blackwell	2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 December 2000*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 October 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *8/27/02*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Specification

The specification is objected to because it does not clearly provide support for the invention's claims. Applicant includes additional descriptions to the point where finding that disclosure intended as support is exceedingly difficult. The specification should be amended to remove this extraneous content.

The specification is also objected to as it does not cite the serial numbers and the filing dates of the related applications listed on pages 1-4.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Independent Claims 18 and 19 are rejected under 35 U.S.C. 101 because they are directed to non-statutory subject matter. Independent Claims 18 and 18 are computer programs per se, and are not tangibly embodied on a computer readable medium.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-6, 13-16, and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Salas et al. (hereinafter Salas, U.S. Patent No. 6,233,600).

In regard to independent Claim 1 (and similarly independent Claims 14-16, and 19), Salas teaches a page builder executing on the client computer for processing the subset of project data from the client database with the template files stored in the client computer to render on the client computer a plurality of HTML pages, representing a portion of a collaborative workspace, the HTML pages enabling the user to perform work on the subset of the project data, for display at the client computer (Col. 16, lines 57-64; compare with Claim 1 (and similarly Claims 14-16, and 19), ***"... preparing at a browser using a legacy editor a form containing hyperlinks that point to web pages"***). Though Salas does not specifically teach *creating a form containing hyperlinks that point to web pages*, Salas does teach dealing with eRooms which are a set of connected HTML pages displayed to a user that displays project-related files, data, and discussion lists. It would have been obvious to one of ordinary skill in the art at the time of invention that with the teaching of Salas, one could have created just about any type of document, including an HTML form document, which likely would

have contained hyperlinks pointing to web pages. Salas allows for the collaborative mechanism to perform such a task locally on a client with legacy editors. Salas also teaches that a background daemon waits for the indicated application to exit, or the document to close, before taking further action (step 606). Once the application has exited, the background daemon determines if the file has been modified (step 508). At this point, Salas does not explicitly state that the file is saved to local storage at said browser. However, assuming that one has created a new file (see Fig. 7, Col. 13, lines 14-26), and one exited the editor, as indicated by Salas above, then one of ordinary skill in the art at the time of invention would have found it obvious that the file would have been saved. Given a background daemon running locally on the client machine is checking for the status of the file, it makes sense that it would more than likely be checking the status of a locally saved file (in other words, not accessing a server and checking file statuses remotely). Hence Salas teaches *saving the file on a client machine.* In addition, Salas also teaches that if the file was modified, then it must be uploaded to the server 14 using HTTP (step 610). The file upload may be done in the background or in the foreground. If done in the foreground the user will be blocked from further work on that file until the upload is complete. Once the upload is complete, the server 14 updates metadata stored in its database 20 that is associated with the file, for example, any edit lock set by the editing user is released (Col. 132, lines 60-67). Also, once the user is finished editing the file, it may be uploaded to the server 14 to allow other users access to it. The user signals that the file should be transmitted to the server 14 by *dragging the file onto an eRoom* displayed by the browser (step 704).

Dropping the file into the displayed eRoom invokes an ActiveX control or a background daemon process which manages the upload of the file to the server 14 (Col. 13, lines 19-26; compare with Claim 1 (and similarly Claims 14-16, and 19), ***"... dragging and dropping said form from local storage into an upload control panel in a user interface to said collaboration space"***).

In regard to dependent Claim 2, Salas does not specifically teach that *said hyperlinks providing access to files including text, graphics, images, sound, and video files*. However, Salas does teach eRooms, which are a set of connected HTML pages displayed to a user that displays project-related files, data, and discussion lists. It would have been obvious to one of ordinary skill in the art at the time of invention that with the teaching of Salas, one could have created just about any type of document, including an HTML form document, which likely would have contained hyperlinks. Those hyperlinks, as it is well known in the art, could have pointed to the claimed file types.

In regard to dependent Claim 3, Salas does not specifically teach that *said web pages residing on servers remote from the server for said collaboration space*. However, Salas does teach eRooms, which are a set of connected HTML pages displayed to a user that displays project-related files, data, and discussion lists. It would have been obvious to one of ordinary skill in the art at the time of invention that with the teaching of Salas, one could have created just about any type of document, including an HTML form document, which likely would have contained hyperlinks. Those hyperlinks, as it is well known in the art, could have pointed to the claimed file types, and could

have existed either on the collaboration server or outside that system on any of a myriad of web servers.

In regard to dependent Claim 4, Salas does not specifically teach that *responsive to user selection of a said hyperlink, downloading linked code; and rendering said linked code into a display at said browser*. However, it is notoriously well known that selecting a hyperlink from a browser usually begins the process of downloading the page and its contents and eventually produces a rendering of the page on the browser.

In regard to dependent Claim 5, Salas teaches a collaborative work environment (*collaboration space*) accessible via a network using a client/server arrangement. The server stores information relating to a project or a set of projects, referred to as a facility (*place*), in a database (20), which can be object-oriented. There can be multiple servers hosting multiple facilities. Each facility is viewed by a user as a directory of eRoom (*room*) pages (Col. 3, lines 14-37). Salas also teaches in Fig. 2 that the structure of one facility in the collaboration space is basically that of a hierarchical, object based (*object model*) file structure each entry providing at least one link to an eRoom front page (24), which in turn may contain other eRoom pages (27), folders containing database objects (28) or files (29) (Col. 3, lines 33-37; compare with Claim 5, ***"... said collaboration space being implemented within an object model selectively including place, room, folder, page, member, form, field, placetype, roomtype, skin, and placebot objects"***).

In regard to dependent Claim 6, Salas teaches that users interact with eRooms by using Web browsers in a traditional manner. That is, users may traverse a hyperlink

to access an eRoom, or users may directly enter a URL address into the browser (Col. 6, lines 40-43; Fig. 4). This compares with Claim 6; ***"... said room object visible as a unique identifier in uniform resource locators (urls)"*** in that Salas implies that one can use a URL to completely specify a document.

In regard to independent Claim 13, Salas teaches that a file (can be an html file, html form, word processing file, etc...) is edited by first selecting the file from the given eRoom, at which point it gets downloaded to the client, or it can already exist on the client's mass storage (hard disk, tape drive, memory), or it could be a newly created file (Col. 12, lines 7-22, 47-67; Col. 13, lines 1-26). Once the file has been downloaded, or if the file was already present in local mass storage, the Watcher launches the application used to edit the file (step 604). The indicated application may be determined using the Object Linking Embedding standard (OLE), the file suffix (three characters in a DOS-based system), or the server 14 may store file metadata which is transmitted together with the file and indicates which application should be used to open and edit the file. If the server 14 stores file metadata, a list of alternate applications may be stored, either on the server 14 or the client workstation 12, so that if a client workstation does not have access to a first application, other applications are specified which may be used (Col. 12, lines 47-59). Compare with Claim 13, ***"... using an hypertext markup language (html) editor"***). Salas does not specifically teach *manually creating an html form with the editor*. However, it is well known that one can create html forms, as well as any other html document using an editor. Further, according to html standards, the content for the form would have needed to be placed within the body

tags of the html document. Salas continues by teaching that the user signals that the file should be transmitted to the server 14 by *dragging the file onto an eRoom* displayed by the browser (step 704). *Dropping the file into the displayed eRoom* invokes an ActiveX control or a background daemon process, which manages the upload of the file to the server 14 (Col. 13, lines 19-26; compare with Claim 13, ***"... uploading said document to said place by dragging and dropping said document to an upload control in a browser user interface to said place"***). Salas fails to teach that ***at said place, replacing head and title tags in said document with place tags"***). However, it would have been obvious to one of ordinary skill in the art at the time of invention to modify the file once uploaded to a collaboration space simply to make the file usable in a collaborative environment.

Claims 7-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Salas in view of Hanson et al. (hereinafter Hanson, U.S. Patent No. 5,956,736).

In regard to dependent Claim 7, Salas does not explicitly teach *using said field object to construct html formatted input fields in forms*. However, Hanson teaches building an HTML document using an object oriented editor and HTML objects (Abstract). Hanson also teaches that having selected an HTML object from either the user or widgets panel of the HTML Palette (500) for inclusion in the collection of objects comprising the current Web document in the page structure panel, a user can edit or modify the properties associated with the object via a context sensitive object editor window. For example, with reference to Fig. 6C, a user clicks on the representation of

the HTML object HTMLText (622) in the page structure panel (608) of Viewer window (600). The user then drags the representation of the HTML object HTMLText (622) and drops it into Object Editor window (610). The object display panel shows an icon and an object name HTMLText (623) representing the HTML object. Furthermore, the properties display panel (614) provides access to the font styles (625), information styles (626) and text string (624) properties associated with the HTML object HTMLText. The user can manipulate any of these properties, as is illustrated by creating the text string "This is a header" in the property display panel (Col. 12, lines 1-18). Hanson also teaches that the present invention allows the Web page created by the object-oriented HTML based editor to be saved as an object rather than a data file in a file system. The object can then be published by a server on the Web as is, or the object can output its contents in a format requested by a client on the Web, e.g., an HTML formatted document (Col. 4, lines 40-45). It would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Salas and Hanson as both inventions teach editors that prepare html documents. Hanson adds the benefit of further defining each component of a web page (such as form input fields) in the form of objects whose attributes can be manipulated.

In regard to dependent Claim 8, Salas fails to specifically teach *defining said form object using said form including imported hyperlinks.* However, Hanson teaches building an HTML document using an object oriented editor and HTML objects (Abstract). Hanson also teaches that having selected an HTML object from either the user or widgets panel of the HTML Palette (500) for inclusion in the collection of objects

comprising the current Web document in the page structure panel, a user can edit or modify the properties associated with the object via a context sensitive object editor window. For example, with reference to Fig. 6C, a user clicks on the representation of the HTML object HTMLText (622) in the page structure panel (608) of Viewer window (600). The user then drags the representation of the HTML object HTMLText (622) and drops it into Object Editor window (610). The object display panel shows an icon and an object name HTMLText (623) representing the HTML object. Furthermore, the properties display panel (614) provides access to the font styles (625), information styles (626) and text string (624) properties associated with the HTML object HTMLText. The user can manipulate any of these properties, as is illustrated by creating the text string "This is a header" in the property display panel (Col. 12, lines 1-18). Hanson also teaches that the present invention allows the Web page created by the object-oriented HTML based editor to be saved as an object rather than a data file in a file system. The object can then be published by a server on the Web as is, or the object can output its contents in a format requested by a client on the Web, e.g., an HTML formatted document (Col. 4, lines 40-45). It would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Salas and Hanson as both inventions teach editors that prepare html documents. Hanson adds the benefit of further defining each component of a web page (such as form input fields) in the form of objects whose attributes can be manipulated.

In regard to dependent Claim 9, Salas fails to teach *parsing said form,*

finding linked files and processing uniform resource locators to generate a form object; saving to a page object said form, said linked files, and said form object; and displaying at said browser said page with said form object in read mode and said form in edit mode. However, Hanson teaches an object comprising a web document being executed by a server, having been previously uploaded to the server from an object-oriented web editor client, and subsequently requested by a client for display (Col. 13, lines 48-57). So, the object (web page) was created by an editor and uploaded to the server as an object. The web page (object) consists of other object (each HTML tag, for example). So, the web object (form) is executed (involving parsing) and tags are converted to whatever format the client requests (e.g., client can request an HTML page, or other characteristics of the object) via handlers

Claims 10-12, and 17-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Salas in view of Kagle (U.S. Patent No. 6,779,153).

In regard to dependent Claim 10, Salas teaches that file download and subsequent upload, if necessary, is managed by a background daemon. Alternatively, file upload and download may be managed by a separately executing program; the only requirement is that the file upload/download application executes separately from the browser application, so that premature exiting of the browser program is handled appropriately by upload/download code (Col. 12, lines 15-22; compare with Claim 10, ***"... downloading said images into a local directory in the same folder as said html page or skin"***). Salas fails to teach *uploading said linked and download images from*

said local directory automatically when said page or skin is uploaded. However, Kagle teaches that pictorial/image information can be entered as a pointer to a locally stored image, which is uploaded with the completed page layout (Col. 6, lines 53-60). Here, a PDA uploads a page layout and an image referred to in the page layout. It would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Salas and Kagle, as both deal with uploading local web content to a server. The benefit of Kagle's teaching is to allow for more efficient uploading by uploading all components of a web site in the same upload session.

In regard to independent Claim 11 (and similarly independent Claims 17-18), Salas teaches that a file (can be an html file, html form, word processing file, etc...) is edited by first selecting the file from the given eRoom, at which point it gets downloaded to the client, or it can already exist on the client's mass storage (hard disk, tape drive, memory), or it could be a newly created file (Col. 12, lines 7-22, 47-67; Col. 13, lines 1-26). Once the file has been downloaded, or if the file was already present in local mass storage, the Watcher launches the application used to edit the file (step 604). The indicated application may be determined using the Object Linking Embedding standard (OLE), the file suffix (three characters in a DOS-based system), or the server 14 may store file metadata which is transmitted together with the file and indicates which application should be used to open and edit the file. If the server 14 stores file metadata, a list of alternate applications may be stored, either on the server 14 or the client workstation 12, so that if a client workstation does not have access to a first application, other applications are specified which may be used (Col. 12, lines 47-59;

compare with Claim 11 (and similarly Claims 17-18), ***"... creating a form in html separate from said place"***). Salas also teaches that the user signals that the file should be transmitted to the server 14 by *dragging the file onto an eRoom* displayed by the browser (step 704). *Dropping the file into the displayed eRoom* invokes an ActiveX control or a background daemon process, which manages the upload of the file to the server 14 (Col. 13, lines 19-26; compare with Claim 11 (and similarly Claims 17-18), ***"... dragging and dropping said form into said place"***). Salas fails to teach ***parsing said form to identify each html field and process uniform resource locators***. However, Kagle teaches in Fig. 8 a process whereby template files are created on a client computer (a PDA in this case). Template files and any images, audio, etc. that are referenced in the template files are uploaded to a server. The server processes the templates and builds an html page. Kagle does not discuss *saving on a page said html in read mode and said form in edit mode"*. However, it would have been obvious to one of ordinary skill in the art at the time of invention to leave the created html page in a read-only mode because it was created using a template. The whole reason for having a template is that it allows one to easily incorporate changes to the web page without having to know the details of html. Likewise, leaving the template file in an edit mode allows for those changes, when they occur to be made in order to regenerate the html file with the changes.

In regard to dependent Claim 12, Salas teaches that file uploading may be done in the background or in the foreground. If done in the foreground the user will be blocked from further work on that file until the upload is complete. Once the upload is

complete, the server 14 updates metadata stored in its database 20 that is associated with the file, for example, any edit lock set by the editing user is released (Col. 132, lines 60-67). Also, once the user is finished editing the file, it may be uploaded to the server 14 to allow other users access to it. The user signals that the file should be transmitted to the server 14 by *dragging the file onto an eRoom* displayed by the browser (step 704). *Dropping the file into the displayed eRoom* invokes an ActiveX control or a background daemon process, which manages the upload of the file to the server 14 (Col. 13, lines 19-26; ***"... importing an original html file which contains an image tag and the related image files into collaboration space"***). Salas fails to teach *parsing said original html file to find linked images; creating a modified html file by modifying said image tag in said html file to refer to the copy of the image file attached to document on the server; saving said original html file and said modified html file on a collaboration space page object.* However, Kagle teaches in Fig. 8 a process whereby template files are created on a client computer (a PDA in this case). Template files and any images, audio, etc. that are referenced in the template files are uploaded to a server. The server processes the templates and builds an html page. Kagle does not discuss *displaying said page object with said original html file in edit mode and said modified html file in read mode.* However, it would have been obvious to one of ordinary skill in the art at the time of invention to leave the created html page in a read-only mode because it was created using a template. The whole reason for having a template is that it allows one to easily incorporate changes to the web page without having to know the details of html. Likewise, leaving the template file in an edit mode

allows for those changes, when they occur to be made in order to regenerate the html file with the changes.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to James H Blackwell whose telephone number is 571-272-4089. The examiner can normally be reached on Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Joseph H Feild can be reached on 571-272-4090. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

James H. Blackwell
01/07/05

JOSEPH FEILD
SUPERVISORY PATENT EXAMINER